

Creating Your Own Game in Excel.

Part 1

Planning and Building a Basic Game.

1)	Introduction	2
2)	What is a Game?	3
3)	Your Game.....	4
3.1)	Deciding What This Game Will Be.....	4
3.2)	Planning the Game – Numbers.	4
3.3)	Planning the Game – Screens.....	4
3.4)	Planning the Game – Flow.	5
4)	Building the Game – Part 1	6
5)	Building the Game – Part 2.....	14
6)	Appendix 1 – Game Code.....	21

1) Introduction

This guide is aimed at people who are familiar with Excel, but have not programmed in it before.

During the course of this 3 part series, we will build a flexible number guessing game. Part 1 covers the planning and a very simple start to the game.

At the end of Part 1 you will have a working Guess the Number game, however it won't be the most user friendly game on the planet. The focus of Part 2 will be to improve the usability of the game, and Part 3 will focus on adding extra features to the game.

Note to more advanced users and programmers:

This is not a guide on how to program. Much of the code in this guide can be written more efficiently. This is a beginners guide on how to make a game in Excel.

Good programming practices will be covered in a later guide. The purpose for the current time is simply to build a game, which will have the secondary purpose of introducing the concept of variables and routines and other programming elements. These concepts will be later expanded upon

2) What is a Game?

I know many of you reading this will want to jump straight in and start making a game. Whilst you can do this for very small basic games, you will get stuck later on if you don't plan it out properly. Even small basic games will turn out better if you plan them.

Hold on a sec, don't walk away just yet. During this tutorial you'll create a very small, basic game. The good part about that is the smaller the game, the less planning you'll have to do.

When you break a game down into what makes it, you can pretty much get right down to two fundamental elements:

1. Numbers.
2. Screens on which to display those numbers.

You might be thinking 'What??? Numbers???'.

What I mean by numbers is that games simply keep track of a whole bunch changing numbers that when interpreted correctly, represent a game.

For example, look at Street Fighter 2. The following elements are all described by numbers:

- Your current health.
- Your opponents' health.
- Your position on the screen (two co-ordinates, x and y).
- Your opponents position on the screen (two co-ordinates, x and y).
- The difficulty level (in SF2 this was represented by stars, 1 through 10).
- Which move you are performing.
- Which move your opponent is performing.
- The time remaining in the round.

In addition to the numbers, there are two main screens which the player sees:

- The Map screen.
- The Fight screen.

Planning your game involves working out what numbers need to be used and tracked in the game, and once that is done, what screens will need to be made in order to display those numbers.

The final element of planning is workout out how the game flows. That is, sequentially, what happens and when. This is normally written in pseudo code, which is not real code, rather a short list of things that need to happen.

3) Your Game.

In order to make a game, you must first decide what game you want to make.

The way I decide what game I want to make is by deciding what sort of game I want to play. I find that if I make games based on what other people want, I don't really get behind the game. On the other hand, if I am making a game I want to play, then I'll get better ideas, put more effort into the code and it'll be better game for it.

3.1) Deciding What This Game Will Be.

For your first project, we're going to build a simple 'Guess the number' game. Excel will think of a number between 1 and 10, and you have to guess what that number is.

As you may have realised in step 1, we are going to plan this out properly. It won't take long as this is a very simple game.

3.2) Planning the Game – Numbers.

Our 'guess the number' game will need to keep track of the following numbers:

The hidden number to be guessed.

This is the number picked by Excel that you will try and guess.

The number entered by the player.

This is the number that the player has typed into Excel to try and guess the hidden number.

3.3) Planning the Game – Screens.

Because this game is very basic, we should only need the one screen for this game.

Game Screen.

All game actions, options and events will take place on this one screen.

3.4) Planning the Game – Flow.

This section describes the flow of the game.

1. Excel picks a random number
2. The player enters a 'guess number'
3. If the guess number is different to the random number, return to step two.
4. If the guess number is the same as the random number, display a message stating the guess is correct and then return to step 1

And the following list describes the development steps to be taken.

1. Start with an empty workbook.
2. Rename a worksheet to something suitable (i.e. 'Guessing Game').
3. Identify a cell where Excel will store a random number.
4. Identify a cell where a player can enter a number.
5. Place a 'Guess Number' button to click on for a player to submit their guess.
6. Place a 'New Game' button to click to start a new game.
7. Write the code to setup a new game and start it.
8. Assign that code to the 'New Game' button
9. Write the code to check to see if the player guess and the random number are the same.
10. Assign that code to the 'Guess Number' button.

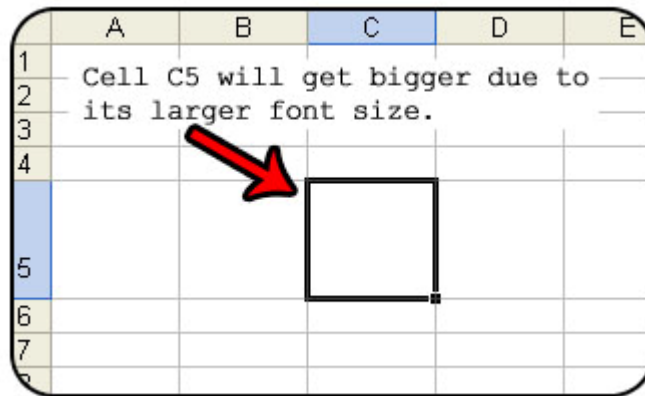
4) Building the Game – Part 1

Now that we have some simple planning done, we are now ready to actually start building the game.

1. If Excel is not already open, fire it up now and get to a blank workbook.
2. Double click on the 'Sheet 1' tab and rename it to 'Guessing Game'.

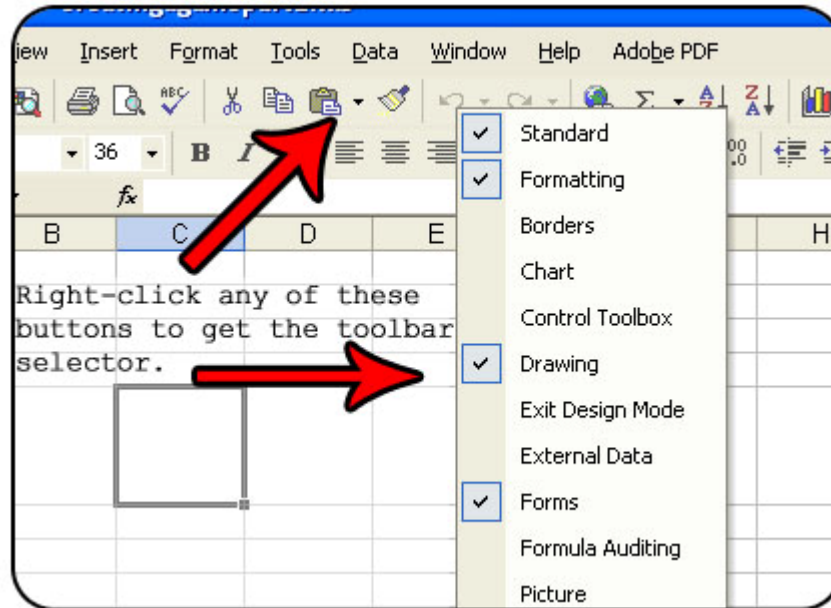


3. Now we need to pick a cell in which Excel will store its random number that the player will try and guess. Cell A1 is as good as any. We don't need to do anything in A1 just yet, just remember that that's the cell we're going to use.
4. Now we will pick a cell that the player will use to enter his/her guess. Let's use cell C5. Select cell C5 and set the font size to 36, centre the text and put a border around it. This makes the players guess box much larger than anything else on the screen.



5. Now we need to place a button on the screen that the player will click to see if their guess is correct or not.

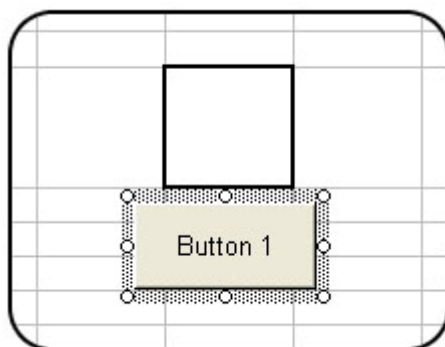
To place a button on the screen, we need the 'forms' toolbar open. To open the forms toolbar, right-click on any of the buttons at the top of the screen, and ensure there is a tick next to the 'Forms' entry.



6. Once you have got the forms toolbar, click on the 'Button' button.



and then drag a button out below cell C5.

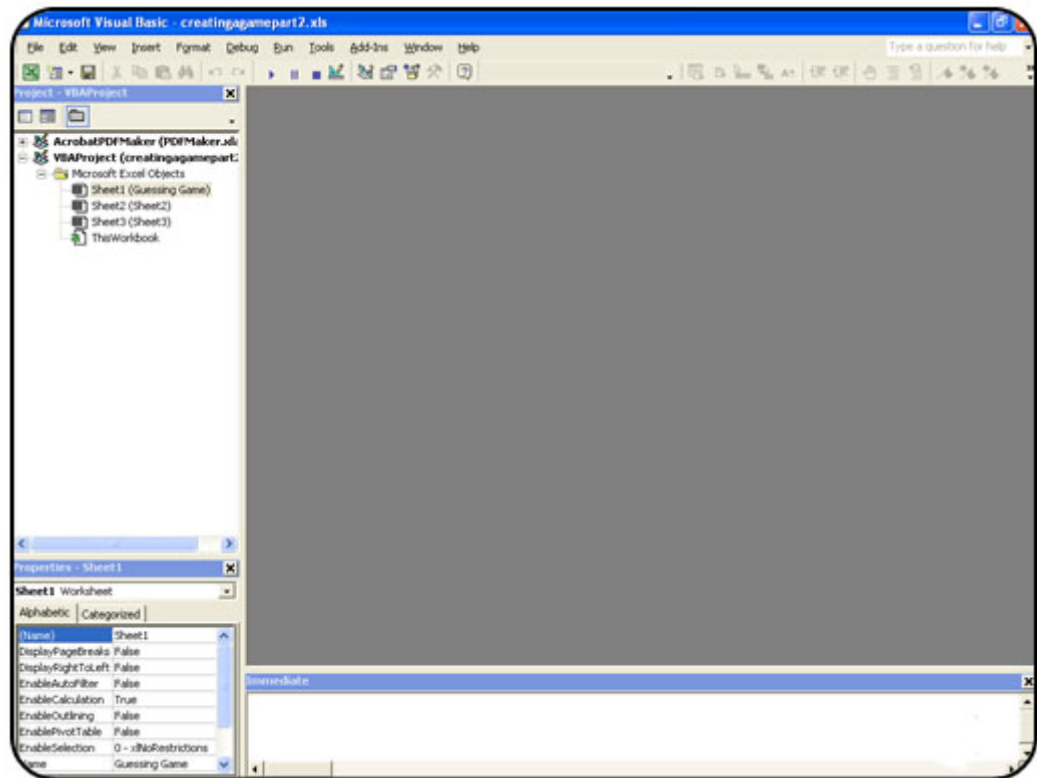


7. Once you have created your button, a window will pop up asking you to assign a macro to the button. Just click 'Cancel'. We'll do this step later.

So far we have a new workbook, with an appropriately named worksheet containing a space for players to guess a number, and a button to click to confirm their guess. We have also identified that we are going to store our random number in cell A1.

Now we are going to write some code. Don't worry if you've never written code before, I'll explain it all as we go along.

8. First we need to open the Visual Basic Editor. This is where all of our code is written. To do this, click on the 'Tools' menu, then 'Macro', then 'Visual Basic Editor' You can also open the Visual Basic Editor by holding the ALT key and pressing the F11 key).
9. You should now have the Visual Basic (VB) editor open. Maximise the window if it is not already.



On the left is the Project Explorer. This lists all of the elements in your workbook. You should be able to see an entry named 'Sheet1(Guessing Game)' in the list. That is your game sheet.

10. Double click on the 'Sheet1(Guessing Game)' entry. This will open the code sheet in the main window (The big grey area will turn white and allow you to type text into it).
11. The first piece of code we are going to type is the code that generates a random number and places it into cell A1.
12. In the white area, type the following code, pressing the Enter key at the end of the line:

You'll notice that the VB editor will have added an additional line of code reading 'End Sub' for you. That is ok. Lets go through what we've done:

This declares a new public sub-routine. You don't need to worry too much about the specifics of what that means right now. More will be covered on that later. What it basically means is 'I'm going to write a chunk of code, and that code will sit between the `Public Sub` and the `End Sub` lines.

This is the name we are going to give our chunk of code (macro). In this case, we are writing the 'startGame' macro.

End Sub

13. So we now have a new macro, however it doesn't do anything yet.



A variable is a container into which we can put a number (or text, or whatever, but in this case it's a number). Our variable needs a name. For the moment, let's call it Bob. If we put the number 5 into Bob, and then later ask the computer 'What's in Bob?' the computer will tell us '5'.

If we then ask the computer again what's in Bob, it will tell us 8 ($5+3=8$).

www.alanchapman.net
Page 9 of 21

use the Enter key to put a few blank lines between the `Public Sub` and the `End Sub` line, and then type the following:

```
dim randomNumber as integer
```

Right, let's explain that:

dim

DIM tells the computer that you want to make a variable. It stands for **Declare In Memory**. This declares your variable to be stored in the computer's memory.

randomNumber

This is the name you give to the variable (like Bob). The name you give a variable should be short (as you will have to type it many times), but also descriptive (so you can tell at a glance what it does).

Variable names must start with a letter, but they can contain numbers. They can NOT contain spaces. Use underscores instead.

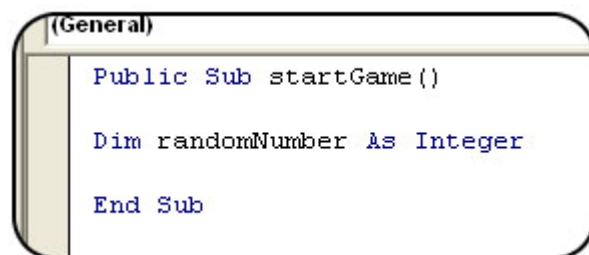
as integer

This tells the computer what type of variable to create. Variables can contain different types of data, i.e. text, whole numbers, decimal numbers, dates and other types. You should use the appropriate type as described below:

text	declare as string	example text
whole number	declare as integer	1 or 2 or 5 or 100
decimal number	declare as single	0.5 or 13.442 or -107.353
date	declare as date	24/10/1980

There are other types of variables but you don't need to know about them yet.

This is what you should have on your screen:



14. So far we have a macro called 'startGame', which creates a variable named 'randomNumber' and that variable can store a whole number (an integer).

The next step is to put a random number in our variable.

To put a number into a variable, you use the following code:

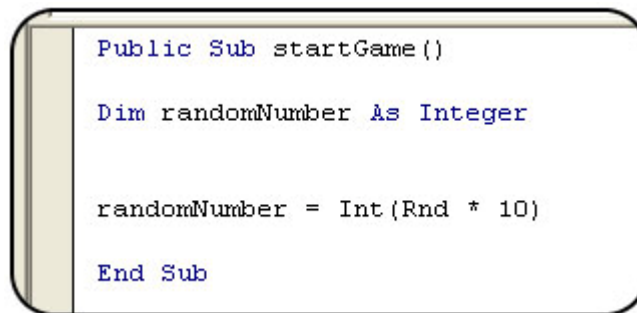
```
randomNumber = 5
```

This code puts the number 5 into the 'randomNumber' variable.

However, we don't want the number 5 in our variable, we want a random number between 1 and 10.

Here is the code to do this. Type this into your macro, and I'll explain it afterwards.

```
randomNumber= int(rnd*10)
```



15. So what does that line of code mean?

randomNumber=

This part of the code says we are about to put something into our randomNumber variable.

int()

This part, INT, stands for integer. The INT command will take any number that is not a whole number (like 2.35) and remove all digits until it is a whole number (an integer, so 2.35 would become 2). the INT command processes any number that is inside of the brackets that follow it.

rnd * 10

the RND command generates a random number between 0 and 1, such as 0.26435, or 0.747892.

The *10 part multiplies the result of the RND command by 10. this has the result of generating a number between 0 and 10, such as 2.6435 or 7.47892.

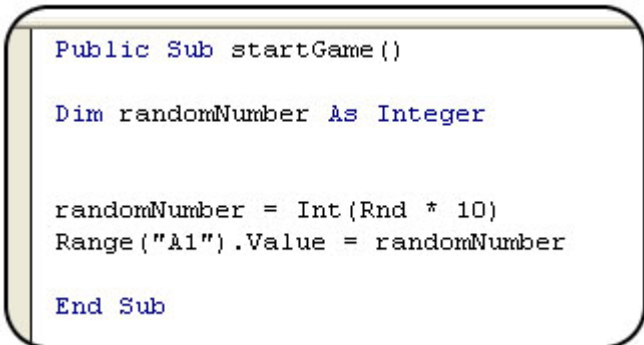
However, the actual line of code read **int(rnd*10)**. So the rnd*10 generates a random number between 0 and 10, such as 6.3952 and then int(6.3952) then reduces this number to a whole number (6).

16. We now have our variable containing a random number. However, the problem with variables is that as soon as this macro has finished running, all variables are destroyed. If you want to use a number later, you should put it into a cell in your workbook.

For this reason, we will be putting the number in this variable into cell A1 so that we can compare it with the number the player enters.

to do this, type the following code underneath the last line we wrote:

```
range("A1").value = randomNumber
```

A screenshot of a VBA code editor window showing the following code:

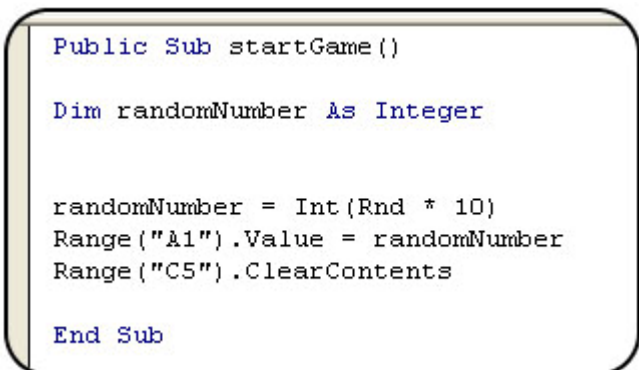
```
Public Sub startGame()  
  
Dim randomNumber As Integer  
  
randomNumber = Int(Rnd * 10)  
Range("A1").Value = randomNumber  
  
End Sub
```

17. This macro is almost complete. It first generates a random number between 1 and 10, and then puts that number into cell A1.

The last thing we need to do is clear whatever number the player has typed into cell C5. This is because we are starting a new game and do not want the previous guess to be part of the new game

To clear cell C5, type the following line of code:

```
range("C5").clearcontents
```

A screenshot of a VBA code editor window showing the following code:

```
Public Sub startGame()  
  
Dim randomNumber As Integer  
  
randomNumber = Int(Rnd * 10)  
Range("A1").Value = randomNumber  
Range("C5").ClearContents  
  
End Sub
```

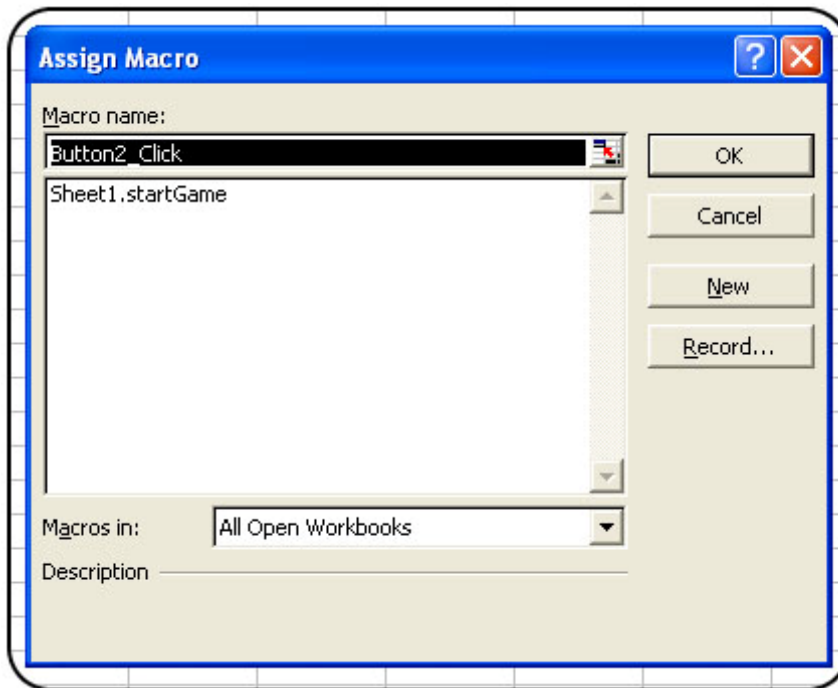
18. Finally, after we start the game, we want Excel to automatically select cell C5, ready for our guess. Type the following line of code:

```
range("C5").select
```

Now return to Excel and create a new button anywhere on the 'Guessing Game' sheet.



When you drag out this button, Excel will pop up a window:



Click on the entry named 'Sheet1.startGame' and then click on the OK button. You have now assigned your startGame macro to the button on the screen. Whenever you click that button, the startGame macro will run.

You can give that button a click and watch random numbers appear in cell A1. Don't worry about the fact that you can see this number. When the game is finished you won't be able to see it (otherwise the game would be a little too easy!).

5) Building the Game – Part 2

The next step in our game is to check to see if what the player has entered into cell C5 matches the number in cell A1.

If the two numbers do not match, a message box should appear telling the player so, and then gives the player another chance at guessing.

If, however the numbers do match, a different message box should appear telling the player they have found the number and won.

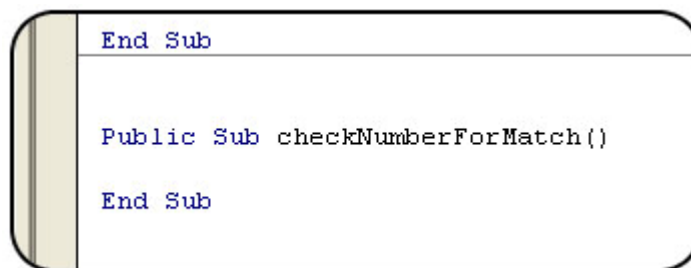
The following steps provide instructions on how to do this:

1. We should now be in Excel. Open the VB Editor again (ALT + F11). We are going to add another macro, which will check the numbers in cells A1 and C5.

Below the `End Sub` line of the `startGame` macro, enter a few blank lines and then type the following code:

```
public sub checkForNumberMatch()
```

The VB Editor should add an `End Sub` line for you.



Use the Enter key to place a few extra lines between the `Public Sub` and `End Sub` of your new macro.

2. In this macro we will need to declare two variables, one for the hidden number, and one for the players guess. Both of which should be whole numbers (integers).

Type the following lines into your new macro, below the `Public Sub` line.

```
dim hiddenNumber as integer  
dim playerGuess as integer
```

```
Public Sub checkNumberForMatch()  
  
    Dim hiddenNumber As Integer  
    Dim playerGuess As Integer  
  
End Sub
```

We have now declared two variables. The next step is to put something into those variables.

3. Before we go throwing numbers around, it is worth us checking to see if the player actually entered a number into cell C5 or not.

Enter a couple of lines after our variable declaration and then type the following code:

Note, there are only 4 lines of code below, The line starting with `response` is one continuous line until "No Guess").

Also, the line starting with `response` should be tabbed in once using the TAB key. The next line should also be indented, and then the last line (`end if`) comes back out. This is not essential, however it helps to keep your code readable.

```
if range("C5").value = "" then  
    response = msgbox("You did not enter a guess.",  
                      vbokonly, "No Guess")  
    exit sub  
end if
```

```
Public Sub checkNumberForMatch()  
  
    Dim hiddenNumber As Integer  
    Dim playerGuess As Integer  
  
    If Range("C5").Value = "" Then  
        response = MsgBox("You did not enter a guess.", vbOKOnly, "No Guess")  
        Exit Sub  
    End If  
  
End Sub
```

Ok, lets explain that.

```
if range("C5").value = "" then
```

This line performs a check on the value in cell C5. If that value turns out to be empty (two quote marks with nothing in them, = "") then Excel will display a message on the screen telling the player they have not entered a

number.

This macro will then stop running (`exit sub`).

If however, the value in C5 is not empty (in other words ,if the check turns out to be false) then Excel will skip all code it encounters next until it find a line reading `end if`, at which point it'll start performing code properly again.

```
response = msgbox("You did not enter a guess.",  
vbokonly, "No guess")
```

This line displays a message box on the screen, giving the player some information. This message box has three pieces of information going into it:

a) `"You did not enter a guess"`

This part tells the message box what the main message displayed in the box should be. This text will appear in the middle of the message box.

b) `vbokonly`

This part tells Excel what buttons to put on the message box. This example places an OK button on the message box. Some other options are `vbYesNo`, `vbYesNoCancel`. There are other options but you don't need to know those yet.

c) `"No guess"`

This part is the text Excel should use in the title bar of the message box. That is the blue bar at the top of the box, similar to the blue bar at the top of word right now that says "Creating Your Own Game in Excel.doc".

```
exit sub
```

This command tells Excel to stop running the current macro, regardless of what else is going on.

```
end if
```

This command ends the `IF` that we started earlier. Code carries on as normal after this point. Only If the `IF` check was true (C5 was empty) will the code between `IF` and `END IF` run.

4. Now that we are sure the player has entered a guess into cell C5, we can go ahead and check if it's the right answer.

First we'll grab both numbers and stick them into our variables:
Enter the following lines of code into your macro:

```
hiddenNumber = range("A1").value  
playerGuess = range("C5").value
```



```
Response = MsgBox("You did not win")
Exit Sub
End If

hiddenNumber = Range("A1").Value
playerGuess = Range("C5").Value

End Sub
```

These lines take the values in cells A1 and C5, and put them into variables that we created earlier.

5. Now we need to compare the two numbers to see if they are the same.

This is how we will structure this check:

```
IF the two numbers are the same
    then do this chunk
    of code that we will
    call block1
OTHERWISE
    do this different code
    which we will call block2
END IF
this code is performed no matter what. This is because
this line of code is outside of the IF - END IF block.
```

This means that when checking to see if the numbers are the same, either the code in block 1 OR the code in block 2 will run, but they can not both run.

If the numbers are the same, run block 1 and then carry on, otherwise, run block 2 and then carry on.

Lets write the actual code for this. Type the following into your macro:

```
if hiddenNumber = playerGuess then
    response=msgbox("Well done. you guessed the right
    number",vbokonly,"You Win")
else
    response=msgbox("I'm sorry, that's not the right
    number.",vbokonly,"Wrong!")
end if
```

```
hiddenNumber = Range("A1").Value
playerGuess = Range("C5").Value

If hiddenNumber = playerGuess Then
    response = MsgBox("Well done. you guessed the right number", vbOKOnly, "You Win")
Else
    response = MsgBox("I'm sorry, that's not the right number.", vbOKOnly, "Wrong!")
End If

End Sub
```

So, if the hidden number, and your guess are the same (one = the other) then Excel will display a message box saying you're right. On the other hand, if the two numbers are different, Excel will display a message box saying you are wrong.

6. That's all of the functionality done. The player types a number into cell C5 and presses ENTER to confirm their entry. Then they click the button below C5 to check their guess.

If their guess is wrong, they will want to try a different number. This means they'll have to re-select cell C5 before entering a new number. Rather than force the player to do this each time, we can automatically re-select cell C5 when checking the guess.

for the last line in the code (well, except the `End Sub` line) enter the following code:

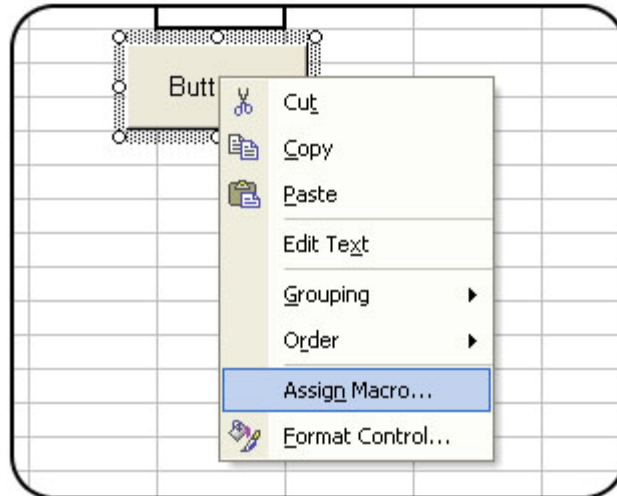
```
range("C5").select
```

This selects cell C5 in Excel.

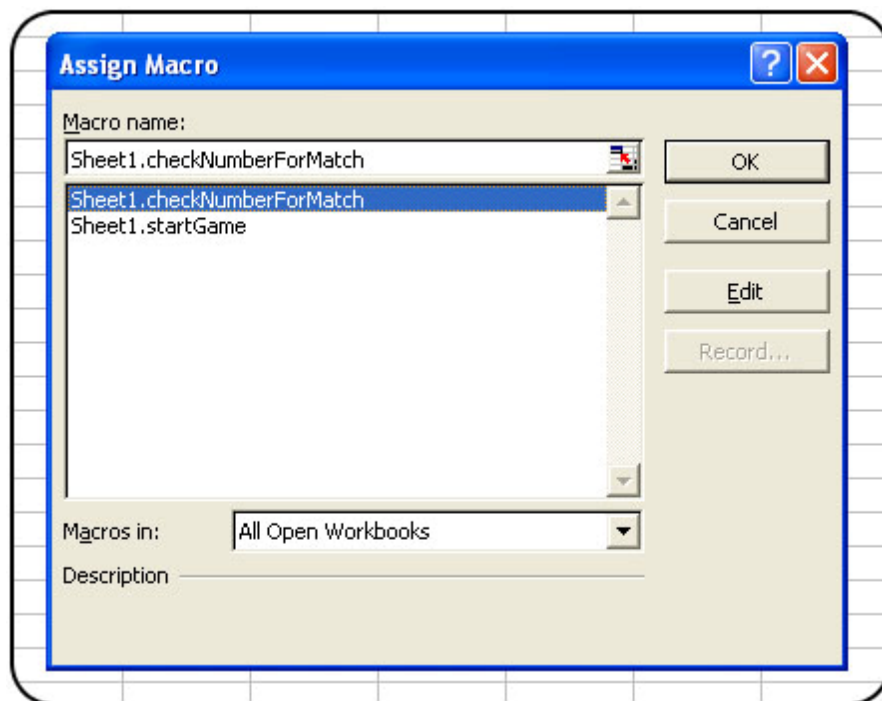
7. Right, we're almost there. That's the code done, now we just need to jump back to Excel and tidy a few things up.

If you're not already in Excel, switch to it now (you can close down the VB editor).

The first thing to do is to assign our newly written macro to a button. Remember the first button we created, just underneath our guess box. Right-click on that button and then select 'Assign Macro' from the drop down menu.



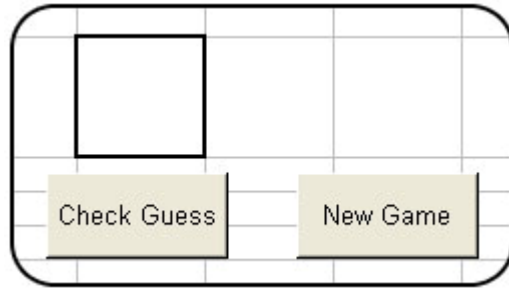
In the box that appears, select `sheet1.checkNumbersForMatch` and click OK. This will tell Excel to run the 'checkNumbersForMatch' macro every time we press this button.



8. There is still a problem with the game in that we can see the hidden number up in cell A1 because it's black text on a white background. To fix this, click on cell A1 and then give the font a white colour.

Once that's done, you can right-click on the buttons and select 'Edit Text' from the menu and change the wording on the buttons. Call them something like "Check Guess" for the guess button and 'New Game' for the new game

button.



9. That's about it for the moment.

Click New Game, and then enter numbers in cell C5 and click the Check Guess button to see if you're right.

If you are, well done. Click on the 'New Game' button to start again. If you're wrong, try a new number.

10. At the moment however, our game is not very good. It doesn't really tell us what's going on, what we should be doing and what it is doing. This makes the game feel unprofessional.

We will fix these issues in part 2 of this guide, and also talk about the plenty of ways we can improve upon this game, such as having Excel tell you if your guess was too high, or too low, or maybe count the number of tries it takes for you to get the right number.

Make sure you save your workbook if you wish to expand upon your game in part 2 as you'll need it as a starting point.

If you've already deleted your game, there is a sample workbook in the second part that will do, however it's better to work with your own file because you created it.

6) Appendix 1 – Game Code

Below you will find the complete code for the game. If for some reason your game is not working, check that your code exactly matches the code here:

```
Public Sub startGame()  
  
Dim randomNumber As Integer  
  
randomNumber = Int(Rnd * 10)  
Range("A1").Value = randomNumber  
Range("C5").ClearContents  
  
End Sub
```

```
Public Sub checkNumberForMatch()  
  
Dim hiddenNumber As Integer  
Dim playerGuess As Integer  
  
If Range("C5").Value = "" Then  
    response = MsgBox("You did not enter a guess.", vbOKOnly,  
        "No Guess")  
    Exit Sub  
End If  
  
hiddenNumber = Range("A1").Value  
playerGuess = Range("C5").Value  
  
If hiddenNumber = playerGuess Then  
    response = MsgBox("Well done. you guessed the right  
        number", vbOKOnly, "You Win")  
Else  
    response = MsgBox("I'm sorry, that's not the right  
        number.", vbOKOnly, "Wrong!")  
End If  
  
range("C5").select  
  
End Sub
```